Tecno Lógicas
ISSN 0123-7799
Vol. 19, No. 36, pp. 77-90
Enero-junio de 2016

Tecnológicas

Programación por demostración de la secuencia de apretar una tuerca admitiendo variaciones en posición de la llave

Programming by demonstration of the sequence of tightening a nut allowing variations in tool position

José G. Hoyos-Gutiérrez[1] y Flavio A. Prieto-Ortiz[2]

Recibido: 31 de julio de 2015,
Aceptado: 11 de noviembre de 2015

Cómo citar / How to cite
J. G. Hoyos-Gutiérrez y F. A. Prieto-Ortiz, "Programación por demostración de la secuencia de apretar una tuerca admitiendo variaciones en posición de la llave", *Tecno Lógicas*, vol. 19, no. 36, pp. 77-90, 2016.

[1] MSc. en Ingeniería Eléctrica, Programa de Tecnología en Instrumentación Electrónica, Universidad del Quindío, Armenia-Colombia, josegabrielh@uniquindio.edu.co

[2] PhD. en Automática, Facultad de Ingeniería, Universidad Nacional de Colombia, Bogotá-Colombia, faprieto@unal.edu.co

Resumen

Se presenta una técnica que permite la programación por demostración de un robot para que ejecute una tarea secuencial o compleja. Se utiliza una combinación de redes de Petri y modelos de mezcla de gaussianas parametrizado en la tarea; con la primera se coordina la secuencia de la tarea, en tanto que la segunda permite variaciones en la posición y orientación de los objetos de la misma. Una técnica de segmentación de tareas, descompone la demostración en subtareas. Con la secuencia de las subtareas, se obtiene una lista de acciones (plan) y con este se genera de manera automática una red de Petri. A la técnica también se le suministran las plantillas modelo de cada subtarea y los modelos de mezcla de gaussianas parametrizados en la tarea de las trayectorias de la subtarea que se quiere que admita variaciones. Una función compara las trayectorias de cada plantilla con las trayectorias repuesta del modelo, y la de mayor similitud indica que en vez de la plantilla, se debe emplear el modelo de mezcla parametrizado. Mediante el uso de un robot de fabricación propia, el cual ejecuta la tarea de tomar, transportar una llave y apretar una tuerca, se ilustra el desempeño de la técnica a través de gráficas.

Palabras clave

Programación por demostración de robots, segmentación de tareas, tareas complejas, modelo de mezcla de gaussianas parametrizado en la tarea, redes de Petri.

Abstract

A technique of programming by demonstration of a robot is proposed. Such a technique allows that a robot execute sequential or complex tasks. It uses a combination of Petri nets and task parameterized Gaussian mixture models. The first one handles the task sequence, while the second one allows variations in the position and orientation of objects involved in the task. Using a segmentation task technique, the demonstration is chunked in subtasks. With the subtasks sequence, an action list or plan is obtained and with this, a Petri net is automatically generate. Models of the templates of each subtasks and task parameterized Gaussian mixture models of the subtask that we want to allow variations are also provide to the technique. A function compare one each of the template trajectory with the task parameterized model response trajectory and the most similar indicate that instead of the template, the task parameterized model is use. Through the use of a homemade robot, which executes the task of tightening a nut, the performance of the technique is illustrated by using figures.

Keywords

Robot programming by demonstration, Task segmentation, Complex tasks, Task parameterized Gaussian Mixture model, Petri nets.

1. INTRODUCCIÓN

La programación por demostración (PpD) es una técnica que permite a un robot aprender la realización de una tarea a través de la demostración de la misma por parte de un humano [1]. Se puede aplicar en entornos poco estructurados, porque permite que un usuario enseñe fácilmente nuevas tareas a un robot sin necesidad de conocimientos en programación. En PpD el robot es entrenado con ejemplos de la tarea, a partir de los cuales el aprende a generalizarla. Las tareas que puede realizar un robot, se pueden dividir en i) habilidades y ii) tareas complejas. Una habilidad describe una acción básica como por ejemplo operaciones de manipulación o transporte; una tarea compleja, es una secuencia de habilidades o subtareas.

Algunos de los retos en PpD en los que se investiga actualmente son: i) Técnicas que permitan generar nuevas trayectorias que guardan similitud con otras que fueron demostradas [2], [3], y ii) Técnicas que permiten programar tareas complejas [4], [5]. En este artículo, se presenta una combinación de los dos temas.

El modelo de mezcla de gausianas GMM (del inglés Gaussian Mixture Models) es un modelo probabilístico, que permite modelar trayectorias en el tiempo con unas pocas gaussianas. Una versión mejorada llamado modelo de mezcla de gaussianas parametrizado en la tarea (en inglés ``Task Parameterized Gaussian Mixture Model'' que en este documento se abrevia como: TPGMM), permite la generación de nuevas trayectorias. Esta mejora fue propuesta por Calinon y otros [3], en la cual, cuando los robots manipulan objetos, sus movimientos pueden depender en gran medida de metas dadas y poses de objetos, las cuales pueden ser definidas a través de marcos de referencia. Concretamente, el movimiento del robot es condicionado por un conjunto de variables de la tarea que representan el sistema coordenado de marcos de referencia relevantes. A cambio de representar cada trayectoria como un modelo diferente, esta técnica, se basa en un solo modelo que abarca las diferentes trayectorias como función de las variables de la tarea, el modelo se fundamenta en las propiedades del producto de gaussianas.

Un ejemplo de tarea compleja, es el manejo de una herramienta por parte de un robot. Por ejemplo, el roscado o desenroscado de una tuerca implica las siguientes subtareas: tomar la herramienta, llevar la herramienta hasta la tuerca con cierta orientación, realizar la operación de roscado o desenroscado, regresar la herramienta a su posición inicial. El manejo de herramientas, permitiría aplicaciones como mantenimiento, ensamble y desensamble. El primer paso, para obtener un programa a partir de las demostraciones de la tarea compleja, es la descomposición de la demostración en subtareas. Para esto se utilizan técnicas de segmentación de la tarea, las cuales dividen la tarea en subtareas. Conociendo la secuencia de las subtareas, se obtiene un plan y con este es posible generar una secuencia de alto nivel que indica las operaciones a realizar. Para lo anterior, se ha usado programación simbólica, gramática de contexto libre, autómatas de estado finito y redes de Petri.

La programación por demostración de la tarea de manejo de la herramienta hace necesario una técnica que coordine la ejecución de las subtareas (ej. red de Petri). Además que permita variar la posición o trayectorias realizadas, al variar la posición de la tuerca, para ello se emplea TPGMM, para generar nuevas trayectorias. El aporte realizado es la combinación de las técnicas de red de Petri y TPGMM en la programación de tareas complejas con generación de nuevas trayectorias.

Otro ejemplo de tarea compleja es el ensamblado de un carro de juguete en madera. Dadas una serie de piezas bien elaboradas, la meta de un ensamble es juntarlas o acoplarlas para producir un objeto que funcione [6]. El proceso total de ensamble se puede dividir en tareas o sub-

ensambles. A partir de la segmentación de la tarea o de extraer la secuencia de acciones, se genera la representación simbólica de la secuencia, se ha usado: gramática como en [7]–[9], grafos como en [10], [11], o modelos ocultos de Markov [5].

En 2005 Zollner y otros [12], a partir de una sola demostración de una tarea compleja, generan un programa de esta. Su ventaja es que requiere una sola demostración, y como desventajas están el que requiere la definición de reglas heurísticas para poder generalizar posiciones, esto último solo se hace para cambios de posición y forma de los objetos, y no para la orientación.

En 2010 Kruger y otros [5] presentan una técnica que detecta primitivas de movimiento de manera no supervisada y con ellas sintetizan tareas. Las primitivas son reconocidas y modeladas usando modelos ocultos de Markov paramétricos (PHMM por sus siglas en inglés), además para la detección de las primitivas, se valen del estado de los objetos. Plantean una dualidad entre el movimiento de los objetos y las acciones realizadas por el humano, basados en esto, usan la segmentación del movimiento de los objetos para separar las acciones del humano en primitivas. Los autores no ahondan en el tema de la obtención de la gramática que se genera a partir de la tarea.

Dantam y otros en 2012 [8], usan gramática de movimiento (motion grammar), para un ensamble simple de una estructura de barras de madera. El sistema convierte los movimientos de los objetos capturados usando un sensor kinect, en códigos que luego el robot reproduce, pero no aborda los problemas de forma de agarre y fuerzas que se presentan, además el sistema aprende una sola secuencia de ensamble.

En 2012 Niekum y otros [4] plantean una técnica para resolver tareas complejas. Primero segmentan la tarea con BP-AR-HMM (del inglés Beta Process Auto Regresive Hidden Markov Models). Luego las subtareas son modeladas usando primitivas de movimiento dinámico (DMP) y vinculadas a un marco de referencia. Una mejora al anterior, fue presentada por los mismos autores en 2013 [11], donde emplearon un autómata de estados finitos para la tarea de enroscar la pata a una mesa. A través de aprendizajes interactivos por parte del ser humano, puede aprender de manera incremental a tomar la pata de la mesa de otras ubicaciones y orientaciones, esto implica adicionar ramas al autómata de estados finitos, las cuales luego son condensadas con las ramas iniciales en sus tramos similares. Las anteriores técnicas, pueden generalizar en cada subtarea, un punto inicial o final, a diferencia de la técnica aquí propuesta que al emplear TPGMM puede tener múltiples parámetros de la tarea.

Las redes de Petri son empleadas por Chang y Kulic en 2013 [10], para tomar, mover, y apilar bloques de madera. La técnica reconoce los objetos y su ubicación (estado), las variaciones de estos estados generan nuevos nodos de la red. Su desventaja es que la generación de los nodos depende de los objetos y no se diseñó para aprender manipulaciones que no involucren objetos.

Abdo y otros en 2013 [13], propone un sistema que genera una representación simbólica de una tarea, basada en las precondiciones y efectos de las acciones. Su ventaja es que puede adaptarse a cambios en la tarea interactuando con el maestro, su desventaja está en que la percepción de las precondiciones debe ser definida manualmente.

En 2014 Martinez y otros [9], emplean una gramática propia y aprendizaje por refuerzo para una tarea de ensamble. Con el movimiento y contacto de los objetos a ensamblar construyen unos grafos, con los cuales entrenan un algoritmo de aprendizaje por refuerzo relacional, según las precondiciones y efectos deseados más adecuados, el algoritmo decide que acción ejecutar. Su ventaja está en que es toleran-

te a variaciones de la percepción de los movimientos, partes no identificadas y errores de inserción.

Cubek y otros en 2015 [14], presentaron una técnica que realiza la tarea de tomar y poner múltiples objetos, esta aprende relaciones conceptuales de forma y color de los objetos con acciones a realizar. Aunque puede generalizar para nuevos objetos, no modelan las trayectorias demostradas, con lo cual no es posible que el robot pueda realizar movimientos que requieran cierta orientación.

El documento presenta la siguiente estructura: en la Sección 2 se presenta la técnica propuesta, la cual obtiene la red de Petri y los modelos de mezcla de gaussianas parametrizados en la tarea. Posteriormente en la Sección 3, se presenta el experimento realizado, sus resultados y discusión. Finalmente se presentan las conclusiones.

2. METODOLOGÍA

En la Fig. 1 se muestran las tres fases que permiten obtener la red de Petri (RdP) [15] a partir de demostraciones de la tarea. En la fase de demostración y segmentación, se obtienen las trayectorias de las articulaciones a partir de una o varias demostraciones de la misma tarea, luego un algoritmo de segmentación de la tarea, permite generar un plan o lista de acciones. En la fase de generación de la red de Petri, el plan es convertido a una lista que describe la red, también en esta fase se genera la matriz de incidencia de la RdP. Por último, en la fase de ejecución de la red, la lista que describe la RdP realiza la secuencia de la tarea.

2.1 Segmentación de la tarea

La segmentación de la tarea permite descomponer la demostración de una tarea compleja, en subtareas. Las subtareas se identifican como los tramos donde existe movimiento en la trayectoria de articulaciones. Cada subtarea tiene una denominación o etiqueta, por ejemplo: agarrar la llave. Para saber a qué etiqueta corresponde un tramo de la tarea completa, se compara este con plantillas obtenidas también por demostración, la comparación se realiza usando CDTW (del inglés Continous Dynamic Time Warping), el cual es un método que calcula la similitud entre dos secuencias.

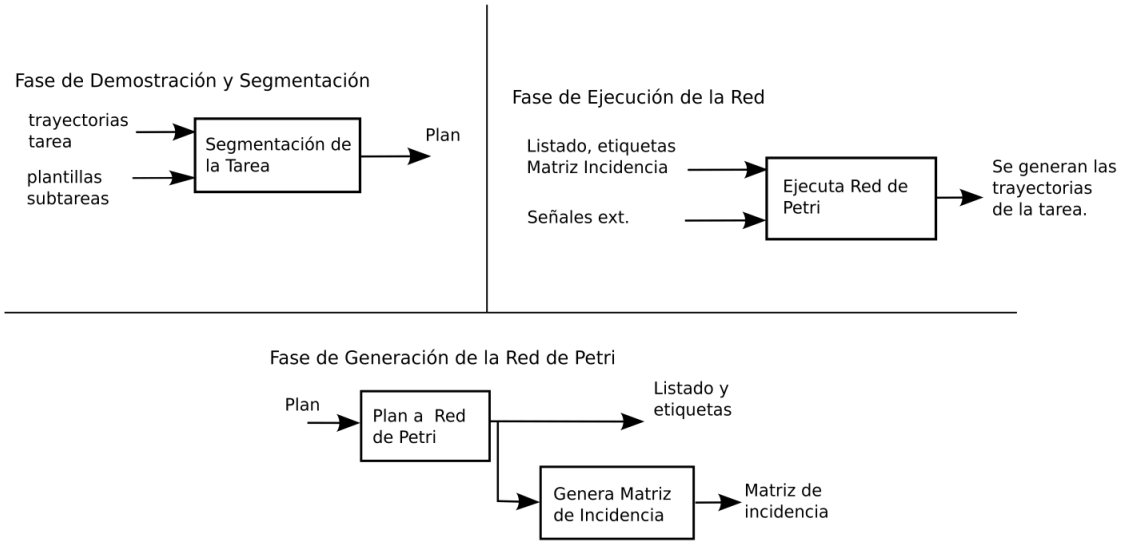


Fig. 1. Fases de la técnica propuesta. Fuente: Autores.

El proceso de segmentación, se divide en dos: i) Obtención de los tramos de trayectorias (subtareas) y ii) Etiquetado de las subtareas. La primera parte a su vez, se realiza a través de los siguientes pasos:

Filtrado de las trayectorias de la tareas.
Cálculo de la suma del cuadrado de las velocidades de las articulaciones.
Obtención de los tramos que sobrepasan el umbral.
Fusión de tramos de corta duración.
Obtención de flancos de inicio y fin de cada subtarea.

Dadas m articulaciones $\boldsymbol{q}$, donde cada articulación tiene n muestras, se filtran usando un filtro de promedio móvil de 4 muestras, para luego calcular la suma del cuadrado de las velocidades con:

$$v_s(k) = \sum_{i=1}^{m} \dot{q}_i^2(k), \quad k = 1, \dots, n. \tag{1}$$

La señal v_s (1) es nuevamente filtrada con un filtro de promedio móvil. Luego, usando un valor umbral, se identifican los tramos donde la suma de velocidades sobrepasa este valor, lo que marca un tramo de movimiento. Este umbral fue prefijado manualmente. En la Fig. 2, se muestra un ejemplo de las trayectorias de cuatro articulaciones y la suma de velocidades obtenida. Ejes q_1 a q_4 corresponden a las cuatro articulaciones del brazo. En la parte inferior, se muestra la suma de velocidades de las articulaciones (línea continua) y en línea a tramos la señal de pulsos resultado de aplicar el umbral. Las zonas que sobrepasan el umbral, son lo que se denomina pulsos (en línea punteada), a los cuales se les realiza una fusión, que agrupa pulsos cercanos y cuyo tiempo sea menor a 20 muestras, dejando un solo pulso, denominado U_s. El paso siguiente es la obtención de los flancos de subida y bajada, con los que se obtiene el inicio (flanco subida) y fin (flanco bajada) de cada subtarea.

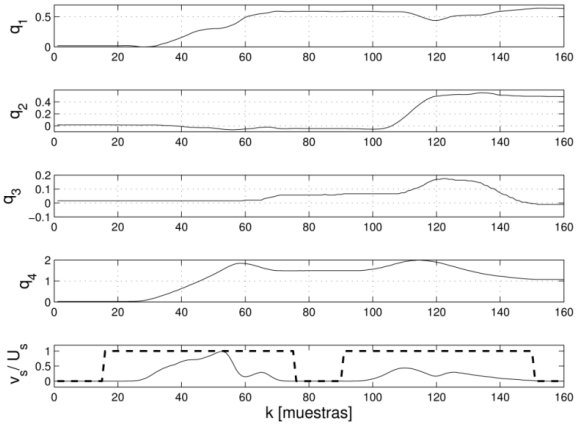


Fig. 2. Trayectorias de articulación de una demostración y suma de velocidades. Fuente: Autores.

2.2 Etiquetado de las subtareas

Para el etiquetado, se obtienen por demostración n_s subtareas plantilla $\boldsymbol{\theta}$, que se etiquetan manualmente y se llevan, todas, a un tamaño de 100 muestras. Por cada subtarea de la tarea completa, se realizan los siguientes pasos:

Cada subtarea de la tarea completa se lleva a un tamaño de 100 muestras.
Se calcula el error por CDTW entre las plantillas y la subtarea.
Se selecciona la plantilla de mayor similitud.

Dado que la mayoría de las subtareas son mayores a 100 muestras, se utiliza la función "spline" para reducirlas y unificalas a un tamaño igual a 100 muestras. Se usa luego CDTW para comparar las subtarea obtenida φ con las plantillas:

$$s_j = cdtw(\varphi, \theta_j), \quad j = 1, \dots, n_s. \tag{2}$$

El índice correspondiente al menor valor del vector s obtenido con (2), suministra la información de que plantilla tiene la mayor similitud con la subtarea de prueba. Para un uso posterior por la red de Petri, a cada plantilla se le estima el modelo GMM.

2.3 Fase de Generación automática de la red de Petri

Una red de Petri [15], es un grafo dirigido que sirve para modelar el funcionamiento de un sistema. Se compone de lugares P, transiciones T y arcos que unen los lugares con las transiciones. La matriz de incidencia contiene información del conexionado de los lugares.

A partir de una lista de etiquetas obtenidas por segmentación de la tarea (plan), una función genera de manera automática dos archivos de texto, el primero es el archivo de equivalencias, que contiene los vínculos de etiquetas con los lugares P y las transiciones T, un ejemplo se muestra a continuación:

P1 : inicio;
P2 : fin_Tomar;
T : Tomar_Herramienta;

El segundo, describe la red de Petri, este archivo se compone de un encabezado, la lista de lugares, la lista de transiciones y la estructura de la red. Por ejemplo la transición *T1* tiene como entrada el lugar *P1* y como salida el lugar *P2*:

Structure:
T1 : (P1) , (P2) ;

Los pesos de los arcos de entrada a una transición, permiten construir la matriz de incidencia previa y los pesos de los arcos de salida de una transición, permiten construir la matriz de incidencia posterior. La matriz de incidencia (3), se arma a partir de las matrices de incidencia previa y posterior:

$$C = C^{+} - C^{-}. \tag{3}$$

2.4 Fase de ejecución de la red

Para la ejecución de la red, se utiliza la regla de evolución del marcado [16]. Dados los elementos de la red (lugares, transiciones), la secuencia de disparos y la matriz de incidencia, la regla de evolución del marcado es:

$$M_k = M_{k-1} + C\mu_k, \tag{4}$$

en la cual M_k (4) es el marcado alcanzado con el vector de disparos μ_k, que consiste en un vector con ceros en las transiciones que no se disparan y uno en la que se dispara en un momento dado. Con lo anterior la red se ejecuta con el Algoritmo 1, donde n_d es el número de disparos y σ (5) es la secuencia de disparos en formato de cadena de caracteres, por ejemplo:

$$\boldsymbol{\sigma = \{'Tomar\ Herramienta', 'Ir\ a\ Tuerca'\}}. \tag{5}$$

Algoritmo 1: Ejecución Red de Petri

Parámetros de entrada: M_0, C, σ

for k=1 to n_d {

$\mu_k \leftarrow$ *convierte_sequencia_a_vector*(σ ,k)

ejecuta_accion(M_{k-1})

$M_k = M_{k-1} + C\mu_k$ }

En su forma simple la función "*ejecuta_accion*" encuentra la acción a ejecutar y con este dato, toma el modelo GMM de cada plantilla en el espacio de articulación y lo reproduce usando GMR (del inglés Gaussian Mixture Regression), generando la trayectoria que ejecuta el brazo robótico.

Como pueden existir diferencias entre los valores finales e iniciales entre las trayectorias de articulación, lo cual crea cambios bruscos, las plantillas se conectaron usando una función "spline" entre los puntos finales de cada plantilla, con los iniciales de la plantilla siguiente.

2.5 Red de Petri con capacidades de variación en la posición de los objetos

Dado que los parámetros de la tarea se estiman en el espacio de la tarea, para la

estimación del modelo TPGMM se transformaron las demostraciones del espacio de articulación al espacio de la tarea, para lo cual se empleó un modelo cinemático directo del robot. Se describe brevemente el modelo de mezcla de gaussianas parametrizado en la tarea y las modificaciones que permiten ahora, a la red de Petri, emplear tanto modelos GMM como modelos TPGMM.

Calinon y otros en 2012 [3], presentan el modelo de mezcla de gaussianas parametrizado en la tarea. Para la estimación del modelo se parte de M demostraciones, N_P parámetros de la tarea $\{\boldsymbol{b}, \boldsymbol{A}\}$ (marcos de referencia de objetos relacionados con la tarea), obteniendo un modelo conformado por coeficientes de mezcla, vector de centros y las matrices de covarianza, una descripción más completa, puede encontrarse Hoyos y otros [17]. En la reproducción del modelo se emplea GMR parametrizado en la tarea, a este se le suministra el o los nuevos parámetros de la tarea, la pose inicial del efector final y el eje de tiempos de las trayectorias, regresando la nueva trayectoria cartesiana.

2.6 Pasando de modelos GMM a modelos TPGMM

Además de los modelos de las plantillas, al programa se le suministran el modelo TPGMM estimado y las demostraciones en el espacio de articulación q correspondientes a las cartesianas usadas para el TPGMM, una función que emplea CDTW, compara ahora las trayectorias de cada plantilla con las trayectorias de articulación demostradas usadas para el TPGMM y la de mayor similitud indica que en vez de la plantilla, se debe emplear el modelo TPGMM (Ver Algoritmo 2). Una nueva versión de la función ejecuta_accion, genera ahora las trayectorias de articulación, empleando los modelos de las plantillas cuando no encuentra similitud, y el o los modelos TPGMM en el caso contrario. La respuesta del modelo TPGMM, es convertida a trayectorias de articulación usando una función de cinemática inversa, que emplea una red neuronal ELM (del inglés Extreme Learning Machine), como fue presentado en [18]. La red ELM se compone de 4 entradas (articulaciones), 200 neuronas en la capa oculta y 4 salidas (pose), con la cual se estima el jacobiano y la cinemática inversa. De manera similar a lo usado para GMM, la conexión entre acciones se realiza con "spline".

Algoritmo 2: Encuentra el índice del modelo a cambiar de GMM a TPGMM

Parámetros de entrada: $\theta_j, \boldsymbol{q}, m, n_s$

Parámetros de salida: índice

$\boldsymbol{err} \leftarrow 0$

for j=1 to n_s

for k=1 to m

$err_j \leftarrow err_j + cdtw(q_k, \theta_{j,k})$

$[minimo, indice] \leftarrow \min(\boldsymbol{err})$

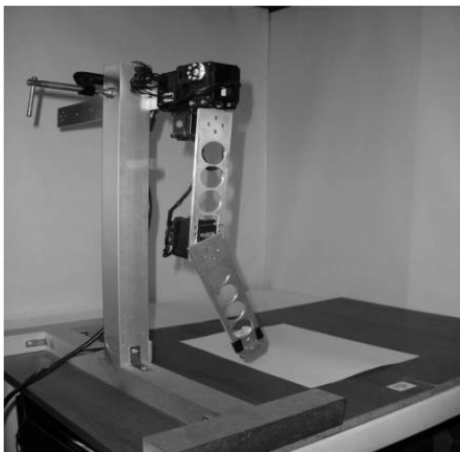
3. RESULTADOS Y DISCUSIÓN

En esta sección se describe el experimento de la tarea de roscado usando un robot de fabricación propia, el roscado es emulado ya que no se emplea una tuerca real. Luego se presentan resultados de la técnica con la red de Petri y GMM, y por último se presentan los resultados de la misma red pero al agregar el modelo TPGMM.

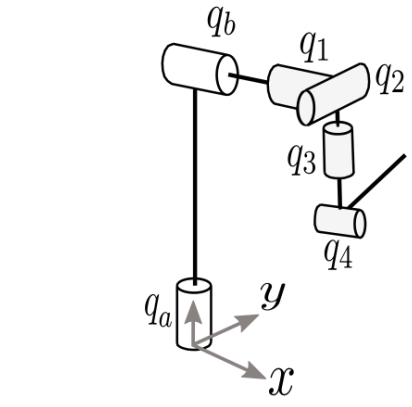
3.1 Detalles del Brazo Robot

En la Fig. 3a se muestra el brazo de cuatro grados de libertad utilizado, este es construido con servomotores dynamixel. Estos servomotores tienen dos modos de funcionamiento: i) como sensor de articulación o ii) como servomotor, en modo sensor es posible capturar los movimientos articulares del brazo, al ser manipulado por una

persona durante la realización de una tarea (aprendizaje kinestático), en modo servomotor se envía desde el computador los valores de articulación, que el servomotor realiza. El conjunto de servomotores, se manejó para ambos modos con ROS Fuerte [19].



(a) Brazo robot



(b) Esquema de grados de libertad

Fig. 3. Robot de 4 Grados de libertad utilizado. Fuente: Autores.

En la Tabla 1, se presentan los valores Denavit-Hartenberg correspondientes a la cinemática directa del brazo robótico empleado en el experimento.

Tabla 1. Parámetros de la cinemática del brazo. Fuente: Autores.

Enlace	θ	d	a	α
L_1	q_a	L_c	0,00	$\pi/2$
L_2	q_b	d_a	0,00	0
L_3	$q_1 + \pi/2$	0,030	0,00	$-\pi/2$
L_4	$q_2 - \pi/2$	0,043	0,00	$\pi/2$
L_5	$q_3 + \pi/2$	L_{b1}	0,02	$\pi/2$
L_6	$q_4 + \pi/2$	0,00	L_{b2}	$\pi/2$

$L_c = 0{,}43\ m;\ \ d_a = 0{,}11\ m;\ L_{b1} = 0{,}167\ m;\ L_{b2} = 0{,}165\ m$

Usando el modelo cinemático del brazo (Tabla 1), y empleando un modelo de seis grados (q_a y q_b no varían), las demostraciones son transformadas del espacio de articulación al espacio de la tarea (x, y, z, α), donde α es un ángulo de orientación.

3.2 Experimento

Como en [17] ya fue mostrada la capacidad de la técnica de TPGMM de lograr la orientación de la llave para casar en la tuerca, en este trabajo se pretende mostrar que es posible emplear TPGMM en conjunto con redes de Petri, para realizar tareas complejas donde pueden variar las posiciones y orientaciones de los objetos.

En la Fig. 4 se muestran imágenes de la secuencia de la tarea de apretar una tuerca, la cual se compone de las siguientes subtareas: Tomar la Herramienta (TH), Ir a la Tuerca (IT), Apretar (AP), Regresar la Herramienta a su posición inicial (RH), Regresar el Efector final a su posición inicial (RE). Se emplearon 20 demostraciones de la tarea completa, descrita por las cinco acciones anteriores.

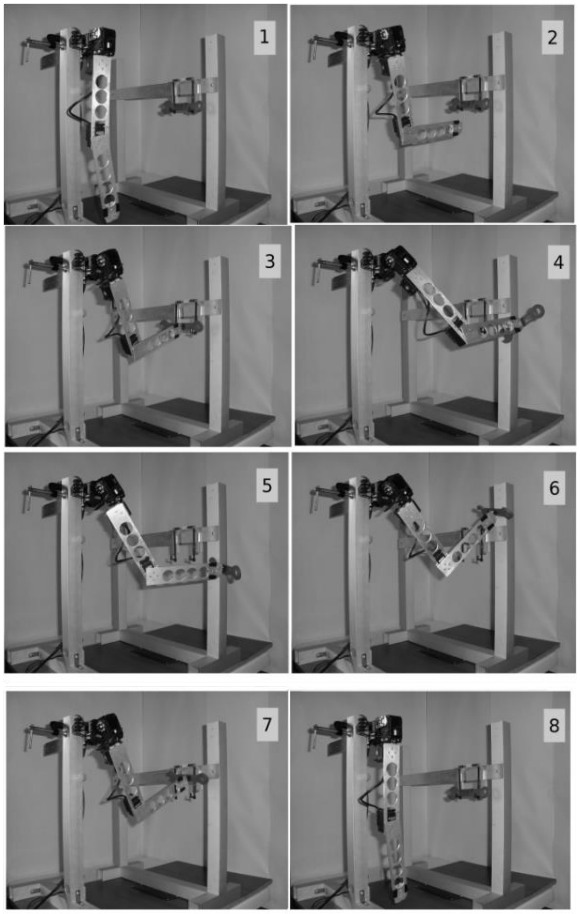


Fig. 4. Imágenes relacionadas con cada subtarea de la tarea de apretar. TH: 1-3, IT: 3,4, AP: 4,5, RH: 6,7, RE: 8. Fuente: Autores

3.3 Resultados con la red de Petri

Las cinco plantillas (TH, IT, AP, RH, RE) que se requieren para la identificación de cada subtarea, se obtuvieron a partir de una de las demostraciones, segmentando de manera manual. En la Tabla 2 se muestran el número de ocurrencias resultado de la identificación de cada subtarea por la técnica, el porcentaje de identificación positivo es del 85%, en la Fig. 5, se muestra un ejemplo del resultado de la segmentación de una de las demostraciones. Donde q_1 y q_3 están en línea continua, q_2 y q_4 en línea punteada, en U_s los cuadrados en color gris y la equis en negro, indican el inicio y fin de una subtarea.

Como la identificación de las subtareas es menor al *100%*, se emplean cuatro demostraciones de la tarea completa, para obtener una secuencia general o plan, para esto una función busca la máxima ocurrencia de la subtarea en una ubicación de la secuencia en las cuatro demostraciones de la tarea.

Tabla 2. Número de ocurrencias en la identificación de las subtareas para la tarea del manejo de la herramienta, para *20* secuencias demostradas. Fuente: Autores

Subtarea	TH	IT	AP	RH	RE	No identif.
TH	20	0	0	0	0	0
IT	0	18	2	0	0	0
AP	0	0	17	3	0	0
RH	0	0	0	17	3	0
RE	0	0	0	2	17	1

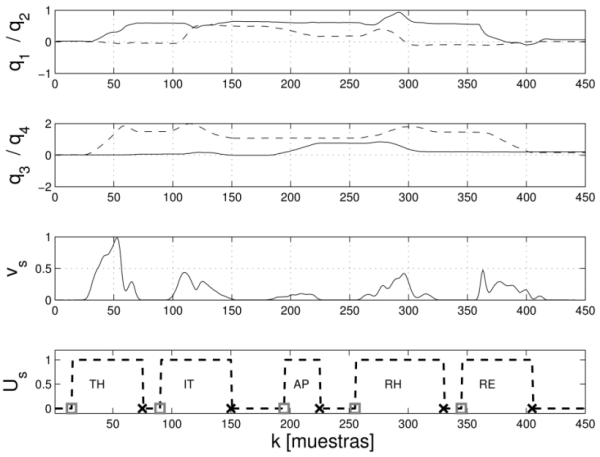


Fig. 5. Articulaciones y resultado de la segmentación de la tarea de apretar una tuerca. Fuente: Autores

Con la secuencia generalizada, se obtiene la red de Petri empleando el Algoritmo 1, presentado en la Sección 2, del cual se presentan listados de la red (Tabla 3a) y etiquetas (Tabla 3b).

Tabla 3a. Red de Petri y lista de etiquetas de la tarea de apretar una tuerca, generada por la técnica. Fuente: Autores

Nodes	T1 : transition ;	Structure
P1 : place (1) ;	T2 : transition ;	T1 : (P1) , (P2) ;
P2 : place ;	T3 : transition ;	T2 : (P2) , (P3) ;
P3 : place ;	T4 : transition ;	T3 : (P3) , (P4) ;
P4 : place ;	T5 : transition ;	T4 : (P4) , (P5) ;
P5 : place ;	T6 : transition ;	T5 : (P5) , (P6) ;
P6 : place ;		T6 : (P6) , (P1) ;

Tabla 3b. Lista de etiquetas de la tarea de apretar una tuerca. Fuente: Autores

P1 : start;	T1 : Tomar_Herr ;
P2: fin_Tomar_Herr;	T2 : Ir_a_T ;
P3 : fin_Ir_a_T;	T3 : Apretar ;
P4 : fin_Apretar;	T4 : Regre_Herr ;
P5 : fin_Regre_Herr;	T5 : Regre_EE ;
P6 : fin_Regre_EE;	T6 : none;

3.4 Resultados de agregar el modelo TPGMM a la red de Petri

Según lo presentado, se agregó la capacidad de variación en la posición del soporte de la llave. En la Fig. 6 se muestran cuatro de las seis demostraciones de la subtarea de tomar la herramienta al variar la posición del soporte de la llave (Fig. 7). Con estas se estima un modelo TPGMM de 4 gaussianas y dos parámetros de la tarea (pose inicial de la punta del brazo y pose final de la punta al tomar la herramienta).

Para obtener los parámetros de la tarea se empleó un procedimiento similar al empleado en [17], en la Fig. 6 se muestran los puntos p_1 y p_2 para el parámetro de la tarea final. En el caso de la estimación p_2 se obtiene tres muestras desplazado del punto inicial o final. Con estos puntos y el procedimiento comentado, se emplean las siguientes ecuaciones:

$$\boldsymbol{b} = \begin{bmatrix} t \\ p_{1x} \\ p_{1y} \\ p_{1z} \\ \alpha \end{bmatrix} \quad \boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & v_{1x} & v_{2x} & v_{3x} & 0 \\ 0 & v_{1y} & v_{2y} & v_{3y} & 0 \\ 0 & v_{1z} & v_{2z} & v_{3z} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

para el cálculo de $\boldsymbol{A}$ usando (6), solo se usan los valores de posición cartesiana.

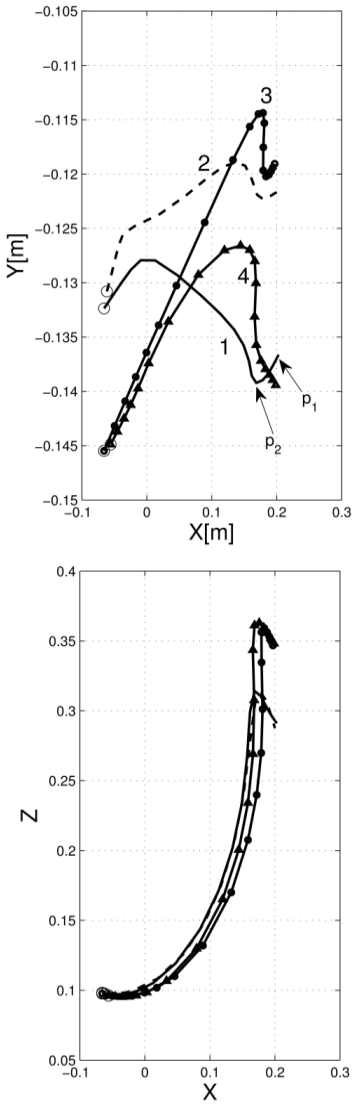


Fig. 6. Demostraciones de la subtarea de tomar la herramienta al variar la posición del soporte de la llave. Fuente: Autores

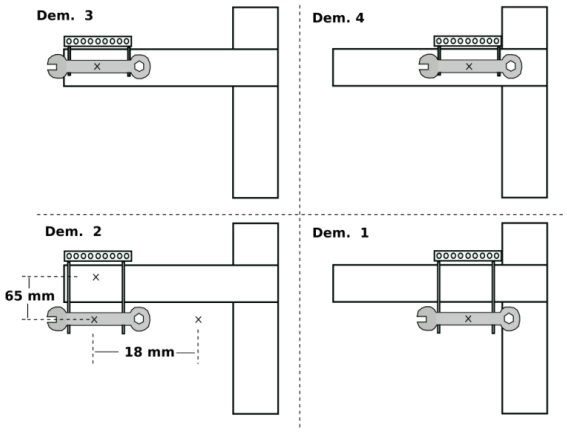


Fig. 7. Diversas posiciones del soporte de la llave. Fuente: Autores

En el caso de reproducción de nuevas trayectorias, el punto p_2, se halla de manera geométrica, dado un punto final p_1, este se traslada una distancia perpendicular al plano *YZ*. Las trayectorias respuesta en línea gruesa, usando el modelo TPGMM, para un parámetro de la tarea central al de las cuatro primeras demostraciones, son mostradas en la Fig. 8.

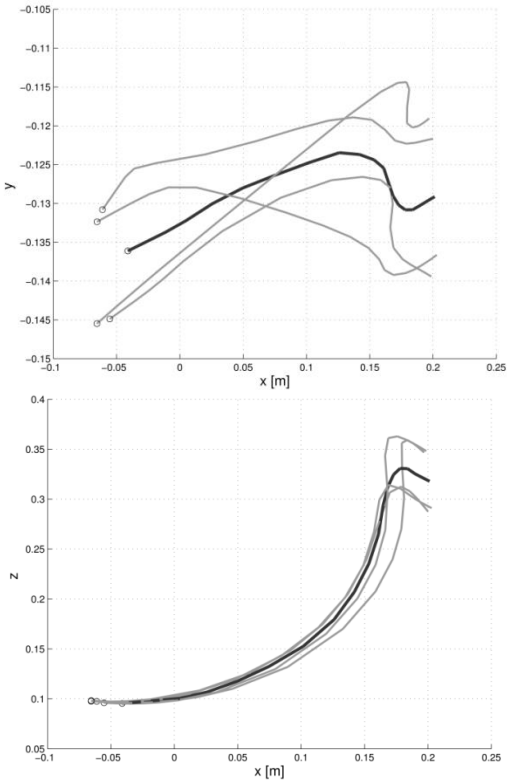


Fig. 8. Trayectoria de reproducción usando el modelo TPGMM (Línea gruesa en negro, demostraciones en gris). Fuente: Autores

Las trayectorias de articulación de la secuencia de la tarea completa para un tramo de *40* muestras y dos posiciones diferentes del soporte, se presentan en la Fig. 9. Línea continua en gris, trayectorias de articulación sin usar TPGMM. Línea negra con triángulos, trayectoria de articulación para la posición del soporte número 2. Línea negra continua, trayectoria de articulación para la posición 4.

En la Fig. 9. es posible observar cómo en la muestra *20* cada valor de articulación es distinto, debido a que la variación de posición de la llave conlleva una variación en el valor de articulación respectivo, de la muestra *20* a la *28* se tiene una línea recta, la cual es la respuesta de la función ``spline" que trata de suavizar el cambio entre subtareas, de la *28* en adelante se tienen las respuestas de los modelos GMM de las plantillas.

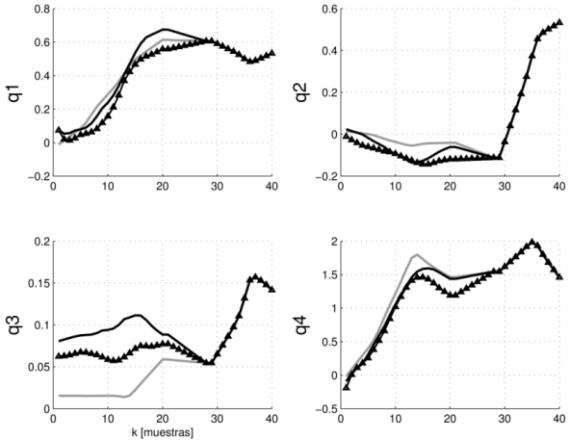


Fig. 9. Trayectorias de articulación generadas al ejecutar la técnica con la red de Petri y agregado el modelo TPGMM. Fuente: Autores

En la Tabla 4 se presenta una lista de otras técnicas existentes, de las cuales solo una generaliza para múltiples parámetros por subtarea y lo hace de manera parcial, ya que emplea la técnica de PHMM, que no responde tan bien como TPGMM, como se mostró en [20]. Además, consideramos que la técnica propuesta en este trabajo es mucho más simple de implementar.

Tabla 4. Técnicas existentes. Fuente: Autores

Autores, Año	Modelado de trayectorias	Generaliza para múltiples parámetros por subtarea
Propuesta	TPGMM	Si
Kruger y otros, 2010	PHMM	Si parcialmente
Niekum y otros, 2012, 2013	DMP	No
Dantam y otros, 2012	No modela	No
Chang y Kulic, 2013	DMP	No
Abdo y otros, 2013	DMP	No
Martinez y otros, 2014	No modela	No
Cubek y otros, 2015	No modela	No

4. CONCLUSIONES

Se presentó una técnica que combina redes de Petri y modelos de mezcla de gaussianas parametrizados en la tarea que permite programar por demostración a un robot una tarea compleja, aunque aquí se presentó la aplicación para el manejo de una herramienta, esta podría ser usada en tarea como ensamblado donde varíen las posiciones o tamaños de los objetos, o en labores domésticas.

Dado que la segmentación de la tarea de una sola demostración no garantiza el cien por ciento en la identificación de las subtareas y obtención de la secuencia o plan, es necesario emplear cuatro demostraciones completas.

A partir del plan se genera la red de Petri en forma de listado, con esta, la matriz de incidencia y los modelos de las plantillas, es posible reproducir la tarea nuevamente. Al cambiar modelos GMM por modelos TPGMM, se logra la flexibilidad necesaria en la ejecución de la tarea ante variaciones de posición de los objetos, aunque solo se agregó un modelo TPGMM, es simple agregar más modelos en el desarrollo de la tarea.

Como trabajo futuro se planea mejorar la forma de agregar el modelo TPGMM, de tal manera que las demostraciones parámetro de entrada a la segmentación de la tarea, contengan las variaciones de posición de los objetos y que además esta sea obtenida empleando técnicas de visión de máquina.

5. REFERENCIAS

[1] B. Aude, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," in *Springer: Handbook of Robotics*, B. Siciliano and O. Kathib, Eds. Springer, 2008, pp. 1371–1394.

[2] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, Oct. 2010.

[3] Sylvain Calinon, Z. Li, T. Alizadeh, Nikos G. Tsagarakis, and D. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *8th IEEE-RAS International Conference on Humanoid Robots*, 2012.

[4] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5239–5246.

[5] V. Kruger, D. Herzog, S. Baby, A. Ude, and D. Kragic, "Learning Actions from Observations," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 30–43, Jun. 2010.

[6] Xiaobu Yuan, "An interactive approach of assembly planning," *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 32, no. 4, pp. 522–526, Jul. 2002.

[7] J. Aleotti, S. Caselli, and M. Reggiani, "Toward programming of assembly tasks by demonstration in virtual environments," in *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003.*, 2003, pp. 309–314.

[8] N. Dantam, I. Essa, and M. Stilman, "Linguistic transfer of human assembly tasks to robots," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 237–242.

[9] D. Martinez, G. Alenya, P. Jimenez, C. Torras, J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater, "Active learning of manipulation sequences," in *2014 IEEE International Conference on*

Robotics and Automation (ICRA), 2014, pp. 5671–5678.

[10] G. Chang and D. Kulic, "Robot task error recovery using Petri nets learned from demonstration," in *2013 16th International Conference on Advanced Robotics (ICAR)*, 2013, pp. 1–6.

[11] S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto, "Incremental Semantically Grounded Learning from Demonstration," in *Robotics: Science and Systems*, 2013.

[12] R. Zollner, O. Rogalla, M. Enrenmann, and R. Dillmann, "Mapping Complex Tasks to Robots: Programming by Demonstration in Real-World Environments," *Adv. Human-Robot Interact.*, vol. 14, pp. 119–136, 2005.

[13] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss, "Learning manipulation actions from a few demonstrations," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1268–1275.

[14] R. Cubek, W. Ertel, and G. Palm, "High-level learning from demonstration with conceptual spaces and subspace clustering," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2592–2597.

[15] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[16] M. Silva, *Las Redes de Petri: en la Automática y la Informática*. AC, 1985.

[17] J. Hoyos-Gutiérrez, C. Peña-Solórzano, C. Garzón-Castro, and F. Prieto-Ortiz, "Hacia el Manejo de una herramienta por un robot NAO usando programación por demostración," *Tecno Lógicas*, vol. 17, no. 33, pp. 65–76, 2014.

[18] J. Hoyos, F. Prieto, C. Pena, E. Morales, and M. Perez-Cisneros, "Reaching New Positions Using an Extreme Learning Machine in Programming by Demonstration," in *2013 Latin American Robotics Symposium and Competition*, 2013, pp. 100–105.

[19] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[20] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 610–616..